Optimal Algorithms for the All-Shortest-Path Problem on a Network

AUTHOR(S):

NAKAMORI, MARIO; IRI, MASAO

数理解析研究所講究録
215巻 1974年 95-126

最短路問題の"最適"算法について

東京大学 工学部 中森真理雄
伊理正夫

"Optimal" Algorithms for the All-Shortest-Path Problem on a Network

By

Mario NAKAMORI and Masao IRI

Faculty of Engineering, University of Tokyo

Abstract

A class of all-shortest-path algorithms expressed as repeated applications of triple-operations is considered. Triple-operations are regarded as operators which induce transformations on distance matrices, and the algebraic structure of the operator semigroup is investigated from the points of view of equivalence relation, partial order, etc. On such investigation, some of the existing algorithms for a complete network are shown to be optimal, and useful formulae for the formal manipulation of algorithms are listed, by means of which the validity of various algorithms is proved. Also, some specially structured networks are discussed and new optimal algorithms for them are proposed.

1. *Introduction*

The all-shortest-path problems on a network are not only of great significance by themselves but also play important rôles as subproblems of many larger ones of network-flow type in operations research. There have been proposed for this kind of problems many methods of solution, "new" proposals still continuing at present.

Expreience shows that many efficient methods belong to the family of those algorithms which are expressed as repeated applications of the so-called triple-operations (or, operations of min-addition pivoting) on distance matrices. In the present paper we aim at establishing a systematic algebraic theory by which to unify the algorithms in this family.

In §2, triple-operations, or repeated applications of triple-operations, are regarded as operators which induce transformations of distance matrices and the operator semigroup is defined. In §3, an algorithm, as well as an operator, is regarded as a systematic enumeration of paths in a network and the correspondence between algorithms and path-sets is established. Based on that correspondence the validity and optimality of an algorithm are discussed. It is shown that any algorithm valid for nonnegative distance matrices is valid for general distance matrices, and that any valid algorithm for an n-node complete network should have at least $n(n-1)(n-2)$ occurrences

of triple-operations. This last proposition leads to the proof of the optimality of many existing algorithms such as the Warshall-Floyd algorithm, the Dantzig algorithm, the Katayama-Watanabe algorithm, etc. Also the algebraic structures of a class of algorithms are investigated from the points of view of equivalence relation, partial order, etc. Useful formulae for the formal manipulation of algorithms are listed, by means of which the validity of various algorithms is proved. In §4, some specially structured networks, such as cascade networks, star networks, etc., are treated, where some of the existing algorithms for such networks are shown to be nonoptimal and new optimal algorithms are proposed.

Most of the results described in the present paper are based on our preliminary studies reported separately in [1], [2] and [3].

2. *The All-Shortest-Path Algorithms*

Let us consider a network G with n nodes. We denote the set of nodes by $N = \{1, 2, \ldots, n\}$. We assume that G has no parallel branches.

We define a *path from* node i_0 *to* node i_r of *length* r in G as a sequence $(i_0, i_1, \ldots, i_{r-1}, i_r)$ of $r + 1$ nodes of G.

If $i_0 = i_r$, we say that the path is *closed*.

For each branch connecting two nodes, say i and j, we denote by d_{ij} the *span* (or *distance*) of the branch *measured from i to j*. [It may be $d_{ij} \neq d_{ji}$ or $d_{ij} < 0$ for some pairs (i, j).] If two distinct nodes i and j are connected by no branch, we set $d_{ij} = d_{ji} = \infty$. For every node i we set $d_{ii} = 0$. We call the sum of d_{ij}'s along a path the *span* (or *distance*) of the path (we assume the span of a path of length 0 to be 0).

In the following *the span of any closed path is assumed to be nonnegative*, since, otherwise, the problem would have no solution. We call a matrix $D = (d_{ij})$ of spans d_{ij}'s satisfying this condition a *distance matrix on G*.

Let $D = (d_{ij})$ be a distance matrix on G. We define the *shortest distance d^{∞}_{ij} from* node i *to* node j *corresponding to D* as the minimum of the spans of all the paths (of any length) from i to j, and call the path(s) giving d^{∞}_{ij} the *shortest path(s) from i to j corresponding to D*. We call the matrix $D^{\infty} = (d^{\infty}_{ij})$ the *shortest-distance matrix corresponding to D*. The *all-shortest-path problem* on a network G is the problem of obtaining the shortest distance matrix $D^{\infty} = (d^{\infty}_{ij})$ corresponding to an arbitrary given distance matrix $D = (d_{ij})$ on G.

Numerous methods of solution have been proposed for the all-shortest-path problem. Most of them are expressed as repeated applications of the so-called triple-operations, where

a *triple-operation* with *pivot* k_0 on *pair* (i_0, j_0) (to be denoted by $\left\langle {k_0 \atop i_0 j_0} \right\rangle$ in abbreviation) of an n by n matrix $A = (a_{ij})$ transforms A into $B = (b_{ij})$ defined as

$$\left.\begin{aligned} b_{i_0 j_0} &= \min\,(a_{i_0 j_0},\ a_{i_0 k_0} + a_{k_0 j_0})\ ; \\ b_{ij} &= a_{ij} \qquad \text{where } (i, j) \neq (i_0, j_0)\ , \end{aligned}\right\} \tag{2.1}$$

or, in ALGOL-like expression, $\left\langle {k_0 \atop i_0 j_0} \right\rangle$ has the effect:

$$a_{i_0 j_0} := \min(a_{i_0 j_0},\ a_{i_0 k_0} + a_{k_0 j_0}) \tag{2.2}$$

for the array $A = (a_{ij})$. Using this kind of expression, some of the well-known methods of solution for the all-shortest-path problem may be described as follows.

Warshall-Floyd method [4]:

```
for k := 1 step 1 until n do
for i := 1 step 1 until n do
for j := 1 step 1 until n do
a[i,j] := min (a[i,j], a[i,k] + a[k,j])
```

Dantzig method [5]:

```
for k := 1 step 1 until n do
begin  for i := 1 step 1 until k do
       for l := 1 step 1 until k do
       a[i,k] := min(a[i,k], a[i,l] + a[l,k]);
       for j := 1 step 1 until k do
       for l := 1 step 1 until k do
```

```
        a[k,j] := min (a[i,j], a[k,l] + a[l,j]);
        for i := 1 step 1 until k do
        for j := 1 step 1 until k do
        a[i,j] := min (a[i,j], a[i,k] + a[k,j])
end
```

Katayama-Watanabe method [6]:

```
for i := 1 step 1 until n do
for j := 1 step 1 until n do
for k := 1 step 1 until min(i, j) do
a[i,j] := min (a[i,j], a[i,k] + a[k,j]);
for i := n step -1 until 1 do
for j := n step -1 until 1 do
for k := max(i, j) step 1 until n do
a[i,j] := min (a[i,j], a[i,k] + a[k,j]);
for i := 1 step 1 until n do
for j := 1 step 1 until n do
for k := min(i, j) step 1 until max(i, j) do
a[i,j] := min (a[i,j], a[i,k] + a[k,j])
```

Cascade method [7], [8]:

```
for i := 1 step 1 until n do
for j := 1 step 1 until n do
for k := 1 step 1 until n do
a[i,j] := min (a[i,j], a[i,k] + a[k,j]);
```

```
for i := n step -1 until 1 do
for j := n step -1 until 1 do
for k := 1 step 1 until n do
a_ij := min (a_ij, a_ik + a_kj)
```

Doubling power method with successive replacement (see, e.g., [9]):

```
for l := 1 step 1 until 3 do
for i := 1 step 1 until n do
for j := 1 step 1 until n do
for k := 1 step 1 until n do
a_ij := min (a_ij, a_ik + a_kj)
```

In the following we shall mean by an *algorithm* any sequence of triple-operations. Thus, an algorithm is expressed as

$$\left\langle {}^{k_t}_{i_t j_t} \right\rangle \cdot \left\langle {}^{k_{t-1}}_{i_{t-1} j_{t-1}} \right\rangle \cdots\cdots \left\langle {}^{k_2}_{i_2 j_2} \right\rangle \cdot \left\langle {}^{k_1}_{i_1 j_1} \right\rangle \quad (t \geqq 0), \qquad (2.3)$$

where the triple-operations are performed successively from right to left. For two algorithms τ and τ' we denote by $\tau \cdot \tau'$ the algorithm in which the algorithm τ' is performed first and then, τ. For an algorithm τ and a distance matrix D, we denote by τD the result of τ operating on D.

We say that an algorithm τ is *valid* for a network G if $\tau D = D^{\infty}$ for every distance matrix on G (which means that we allow "invalid" algorithms also as algorithms).

We define the *complexity* of an algorithm as the number of occurrences of triple-operations in the algorithm, i.e. the value of "t" in (2.3). We denote the algorithm of complexity 0 by ε, which induces the identity transformation on distance matrices.

We say that a valid algorithm for G is *optimal* if its complexity is the smallest of all the valid algorithms for G.

3. *The Operator Semigroup and Path-Sets*

3.1. *Enumeration of paths by an algorithm.* We begin with the investigation into algorithms from the point of view of enumeration of paths. We call a set of paths a *path-set*. We call the subset of a path-set π, which consists of the paths from node i to node j in π, the (i, j)-*component* of π and denote it by π_{ij}. Sometimes we denote a path-set π in a matrix form as $\pi = (\pi_{ij})$. We denote by $P = (P_{ij})$ the path-set of all the paths in G and by $P^{(r)} = (P_{ij}^{(r)})$ the path-set of all the paths of length not exceeding r in G.

For two paths $p = (i_0, i_1, \ldots, i_{r-1}, i_r)$ and $q = (j_0, j_1, \ldots, j_{s-1}, j_s)$, we define the *concatenation* $p \square q$ as the path $(i_0, i_1, \ldots, i_{r-1}, i_r, j_1, \ldots, j_{s-1}, j_s)$ if $i_r = j_0$ (if $i_r \neq j_0$, $p \square q$ is undefined). For two path-sets π and π', we define the concatenation as $\pi \square \pi' = \{p \square p' \mid p \in \pi, p' \in \pi'\}$. For a path

$p = (i_0, i_1, \ldots, i_{r-1}, i_r)$ and a path-set $\pi = (\pi_{ij})$, we define the path-set $p[\pi]$ by $p[\pi] = \pi_{i_0 i_1} \square \pi_{i_1 i_2} \square \cdots \square \pi_{i_{r-1} i_r}$ if $r \geqq 1$, or by $p[\pi] = \{p\}$ if $r = 0$. For path-sets π and π', we define the *link-product* $\pi \uparrow \pi'$ by $\pi \uparrow \pi' = \bigcup_{p \in \pi} p[\pi]$.

To each algorithm τ we assign a path-set $\rho(\tau)$ as follows:

$$\left.\begin{aligned} &\rho(\varepsilon) = P^{(1)} \ ; \\ &\rho(\langle {}^{k}_{i\ j} \rangle) = P^{(1)} \cup \{(i, k, j)\} \ ; \\ &\rho(\tau \cdot \tau') = \rho(\tau) \uparrow \rho(\tau') \ . \end{aligned}\right\} \qquad (3.1)$$

The intuitive interpretation of ρ is given by the following theorem, which is easy to prove.

Theorem 1. *For a distance matrix D and an algorithm τ, each (i, j)-entry of the resultant matrix τD is equal to the minimum of the spans along all the paths from node i to node j in the path-set $\rho(\tau)$.*

3.2. *Validity of an algorithm.* A path is said to be *elementary* if the nodes on the path are distinct. We denote by $\rho^{\varepsilon}(\tau)$ the path-set of all elementary paths in the path-set $\rho(\tau)$. We say that a path is *conductive* if each pair of neighboring nodes in the path is connected by a branch.

It is obvious that the shortest path from a node to another corresponding to any distance matrix is elementary and conductive.

It is also obvious that any elementary and conductive path from a node to another may be the shortest path corresponding to some distance matrix. Hence follows the

Theorem 2. *An algorithm* τ *is valid for a network* G *if and only if all the elementary conductive paths in* G *are contained in* $\rho^{\varepsilon}(\tau)$.

3.3. *Optimality of an algorithm.* A network is said to be *complete* if every pair of different nodes is connected by a branch.

In a complete network G, all the elementary paths (and, therefore, all the elementary paths of length 2) are conductive. If a triple-operation $\left\langle {}_{i}^{k}{}_{j} \right\rangle$ (i, j, k being distinct) did not occur in an algorithm τ, then the elementary conductive path (i, k, j) would not be contained in $\rho^{\varepsilon}(\tau)$. Hence we have

Theorem 3. *For every triple* (i, j, k) *of distinct nodes* i, j *and* k, *the triple-operation* $\left\langle {}_{i}^{k}{}_{j} \right\rangle$ *occurs at least once in any valid algorithm for a complete network.*

and

Theorem 4. *Any valid algorithm for a complete* n*-node network contains at least* $n(n-1)(n-2)$ *occurrences of triple-operations.*

Theorem 4 shows that the methods by R. W. Floyd [4], G. B.

Dantzig [5], and H. Katayama and H. Watanabe [6], are optimal for a complete network in this sense.

It is important to note that the above-mentioned fact is independent of whether the distance matrix under consideration is nonnegative or not, i.e.

Theorem 5. *If an algorithm transforms every nonnegative distance matrix on a network (not necessarily complete) into the corresponding shortest distance matrix, then the algorithm is valid for that network, i.e. it transforms every distance matrix (not necessarily nonnegative) on the network into the corresponding shortest distance matrix.*

and

Theorem 6. *On a complete n-node network no algorithm which consist of less than* $n(n-1)(n-2)$ *triple-operations (therefore, invalid for a complete n-node network) transforms every nonnegative distance matrix into the corresponding shortest distance matrix.*

Recently, A. J. Hoffman and S. Winograd [10] devised a method by which only $O(n^{5/2})$ additions and subtractions are needed. However, as for the operations of comparison the method is identical with that of Floyd, i.e. it requires about n^3 comparisons. Moreover, the method is not an algorithm in our sense, since it makes use of the combinations of addition/subtraction and comparison in a form different from triple-operations.

3.4. *The operator semigroup.* Let π and π' be path-sets of elementary paths in a network G. If all the conductive paths in π are contained also in π', we write $\pi \subseteq \pi'$ (C). Let τ and τ' be algorithms. We write $\tau \leqq \tau'$ if $\rho^{\varepsilon}(\tau) \supseteq \rho^{\varepsilon}(\tau')$ (C). We say that τ and τ' are *equivalent* and write $\tau \equiv \tau'$ if $\tau \leqq \tau'$ and $\tau' \leqq \tau$. We write $\tau < \tau'$ if $\tau \not\equiv \tau'$ and $\tau \leqq \tau'$. We say that τ and τ' are *incomparable* and write $\tau \nsim \tau'$ if neither $\tau \leqq \tau'$ nor $\tau' \leqq \tau$.

The quotient of the set of algorithms by the equivalence $\equiv$ defined above forms a semigroup, which we shall call the *operator semigroup*. Each member of the operator semigroup (i.e. each equivalence class of algorithms) is called an *operator*.

Let us examine some properties of the operator semigroup.

To begin with, we observe that

$$\left\langle {}^{k}_{i\ i} \right\rangle \equiv \left\langle {}^{k}_{k\ j} \right\rangle \equiv \left\langle {}^{k}_{i\ k} \right\rangle \equiv \varepsilon \quad \text{for every } i, j, k \text{ of } N, \tag{3.2}$$

and

$$\left\langle {}^{k}_{i\ j} \right\rangle \cdot \left\langle {}^{k}_{i\ j} \right\rangle \equiv \left\langle {}^{k}_{i\ j} \right\rangle \quad \text{for every } i, j, k \text{ of } N. \tag{3.3}$$

The commutativity relations between triple-operations are summarized in Table 1, where $\left\langle {}^{k}_{i\ j} \right\rangle = \tau$ and $\left\langle {}^{k'}_{i'j'} \right\rangle = \tau'$. The inequalities $\leqq$ in Table 1 hold strictly (i.e. as $<$) if the underlying network G is complete.

<table>
<tr><td></td><td colspan="3">$j = j'$</td><td colspan="3">$j \neq j'$</td></tr>
<tr><td rowspan="3">$i = i'$</td><td colspan="3" rowspan="3">$\tau\cdot\tau' \equiv \tau'\cdot\tau$</td><td></td><td>$k' = j$</td><td>$k' \neq j$</td></tr>
<tr><td>$k = j'$</td><td>$\tau\cdot\tau' \equiv \tau'\cdot\tau$</td><td>$\tau\cdot\tau' \underset{=}{\leq} \tau'\cdot\tau$</td></tr>
<tr><td>$k \neq j'$</td><td>$\tau\cdot\tau' \underset{=}{\geq} \tau'\cdot\tau$</td><td>$\tau\cdot\tau' \equiv \tau'\cdot\tau$</td></tr>
<tr><td rowspan="3">$i \neq i'$</td><td></td><td>$k' = i$</td><td>$k' \neq i$</td><td colspan="3" rowspan="3">$\tau\cdot\tau' \equiv \tau'\cdot\tau$</td></tr>
<tr><td>$k = i'$</td><td>$\tau\cdot\tau' \equiv \tau'\cdot\tau$</td><td>$\tau\cdot\tau' \underset{=}{\leq} \tau'\cdot\tau$</td></tr>
<tr><td>$k \neq i'$</td><td>$\tau\cdot\tau' \underset{=}{\geq} \tau'\cdot\tau$</td><td>$\tau\cdot\tau' \equiv \tau'\cdot\tau$</td></tr>
</table>

Table 1. Commutativity relations between triple-operations $\tau = \left\langle {k \atop i\ j} \right\rangle$ and $\tau' = \left\langle {k' \atop i'j'} \right\rangle$.

3.5. *Operators of practical use.* From the fundamental properties of the operator semigroup shown in §3.4, we see that

$$\left\langle {k \atop S} \right\rangle \overset{\mathrm{d}}{=} \prod_{(i,j)\in S} \left\langle {k \atop i\ j} \right\rangle \qquad (k \in N\ ;\ S \subseteq N \times N)$$

has its unique meaning as an operator, where the right-hand side means the "."-product of the triple-operations of the form $\left\langle {k \atop i\ j} \right\rangle$ $[(i, j) \in S]$, the order being unimportant. It is easy to prove that

$$\left\langle {k \atop R} \right\rangle \cdot \left\langle {k \atop S} \right\rangle \equiv \left\langle {k \atop R \cup S} \right\rangle \qquad \begin{array}{l}\text{for every } k \in N \text{ and} \\ \text{every } R,\ S \subseteq N \times N.\end{array} \qquad (3.4)$$

The following symbols for subsets of $N \times N$ will be used.

$$\left.\begin{aligned} I(k) &= \{(i, j) \in N \times N \mid i \leqq k \leqq j\}, \\ II(k) &= \{(i, j) \in N \times N \mid i \leqq k, j \leqq k\}, \\ III(k) &= \{(i, j) \in N \times N \mid j \leqq k \leqq i\}, \\ IV(k) &= \{(i, j) \in N \times N \mid k \leqq i, k \leqq j\}. \end{aligned}\right\} \tag{3.5}$$

By an elementary consideration we can prove the commutativity relations on a complete network in Table 2 between operators of the form $\langle {}^{k}_{R(k)} \rangle$ and $\langle {}^{k'}_{S(k')} \rangle$ ($R, S = I, II, III$ or IV), where the commutativity relation between $\langle {}^{k}_{I(k)} \rangle$'s or $\langle {}^{k}_{III(k)} \rangle$'s, respectively, follows from the fact that

$$\langle {}^{k}_{i\ j} \rangle \cdot \langle {}^{k}_{k'j} \rangle \cdot \langle {}^{k'}_{i\ j} \rangle \cdot \langle {}^{k'}_{i\ k} \rangle \equiv \langle {}^{k'}_{i\ j} \rangle \cdot \langle {}^{k'}_{i\ k} \rangle \cdot \langle {}^{k}_{i\ j} \rangle \cdot \langle {}^{k}_{k'j} \rangle$$

$$\text{for every } i, j, k \text{ and } k' \in N, \tag{3.6}$$

which can be examined directly.

We define

	$S = I$	$S = II$	$S = III$	$S = IV$
$R = I$	$\tau\cdot\tau' \equiv \tau'\cdot\tau$	$\tau\cdot\tau' \not< \tau'\cdot\tau$	$\tau\cdot\tau' \equiv \tau'\cdot\tau$	$\tau\cdot\tau' \equiv \tau'\cdot\tau$
$R = II$	$\tau\cdot\tau' \equiv \tau'\cdot\tau$	$\tau\cdot\tau' < \tau'\cdot\tau$	$\tau\cdot\tau' \equiv \tau'\cdot\tau$	$\tau\cdot\tau' \equiv \tau'\cdot\tau$
$R = III$	$\tau\cdot\tau' \equiv \tau'\cdot\tau$	$\tau\cdot\tau' \not< \tau'\cdot\tau$	$\tau\cdot\tau' \equiv \tau'\cdot\tau$	$\tau\cdot\tau' \equiv \tau'\cdot\tau$
$R = IV$	$\tau\cdot\tau' \not< \tau'\cdot\tau$	$\tau\cdot\tau' \not< \tau'\cdot\tau$	$\tau\cdot\tau' \not< \tau'\cdot\tau$	$\tau\cdot\tau' > \tau'\cdot\tau$

Table 2. Commutativity relations on a complete network between operators $\tau = \langle {}^{k}_{R(k)} \rangle$ and $\tau' = \langle {}^{k'}_{S(k')} \rangle$, where $k < k'$.

$$\left.\begin{aligned}\Gamma_r^F &\triangleq \left\langle {}^{n}_{R(n)}\right\rangle\cdot\left\langle {}^{n-1}_{R(n-1)}\right\rangle\cdots\cdots\left\langle {}^{2}_{R(2)}\right\rangle\cdot\left\langle {}^{1}_{R(1)}\right\rangle\ ,\\ \Gamma_r^B &\triangleq \left\langle {}^{1}_{R(1)}\right\rangle\cdot\left\langle {}^{2}_{R(2)}\right\rangle\cdots\cdots\left\langle {}^{n-1}_{R(n-1)}\right\rangle\cdot\left\langle {}^{n}_{R(n)}\right\rangle\ ,\end{aligned}\right\} \qquad (3.7)$$

where we set r = 1, 2, 3 or 4, according as $R = I$, II, III or IV, respectively. If $\left\langle {}^{k}_{R(k)}\right\rangle$'s are commutative for different k's, we write simply as Γ_r (cf. Table 2). We sometimes use such notations as Γ^F_{rs}, Γ^B_{rs} or Γ^F_{rst}, Γ^B_{rst} to mean a product of operators $\left\langle {}^{k}_{R(k)\cup S(k)}\right\rangle$'s or that of $\left\langle {}^{k}_{R(k)\cup S(k)\cup T(k)}\right\rangle$'s, whose rigorous definition will be evident. Thus the following notations have their unique meanings as operators.

$$\Gamma_1,\quad \Gamma_3,\quad \Gamma_{13},$$

$$\Gamma^B_2,\quad \Gamma^F_2,\quad \Gamma^B_{12},\quad \Gamma^F_{12},\quad \Gamma^B_{23},\quad \Gamma^F_{23},\quad \Gamma^B_{123},\quad \Gamma^F_{123},$$

$$\Gamma^B_4,\quad \Gamma^F_4,\quad \Gamma^B_{34},\quad \Gamma^F_{34},\quad \Gamma^B_{14},\quad \Gamma^F_{14},\quad \Gamma^B_{134},\quad \Gamma^F_{134}.$$

The intuitive interpretation of these operators are given in Fig. 1, where the shaded area indicates the set of the pairs of lower indices of triple-operations with the pivot indicated by a small circle and the arrow indicates the movement of the pivot.

From the properties in Table 2 we can obtain various relations among the above operators. For example, we have

$$\left.\begin{aligned}&\Gamma_{13} \equiv \Gamma_1\cdot\Gamma_3 \equiv \Gamma_3\cdot\Gamma_1,\\ &\Gamma^B_{12} \equiv \Gamma_1\cdot\Gamma^B_2, \qquad \Gamma^F_{12} \equiv \Gamma^F_2\cdot\Gamma_1,\end{aligned}\right\} \qquad (3.8)$$

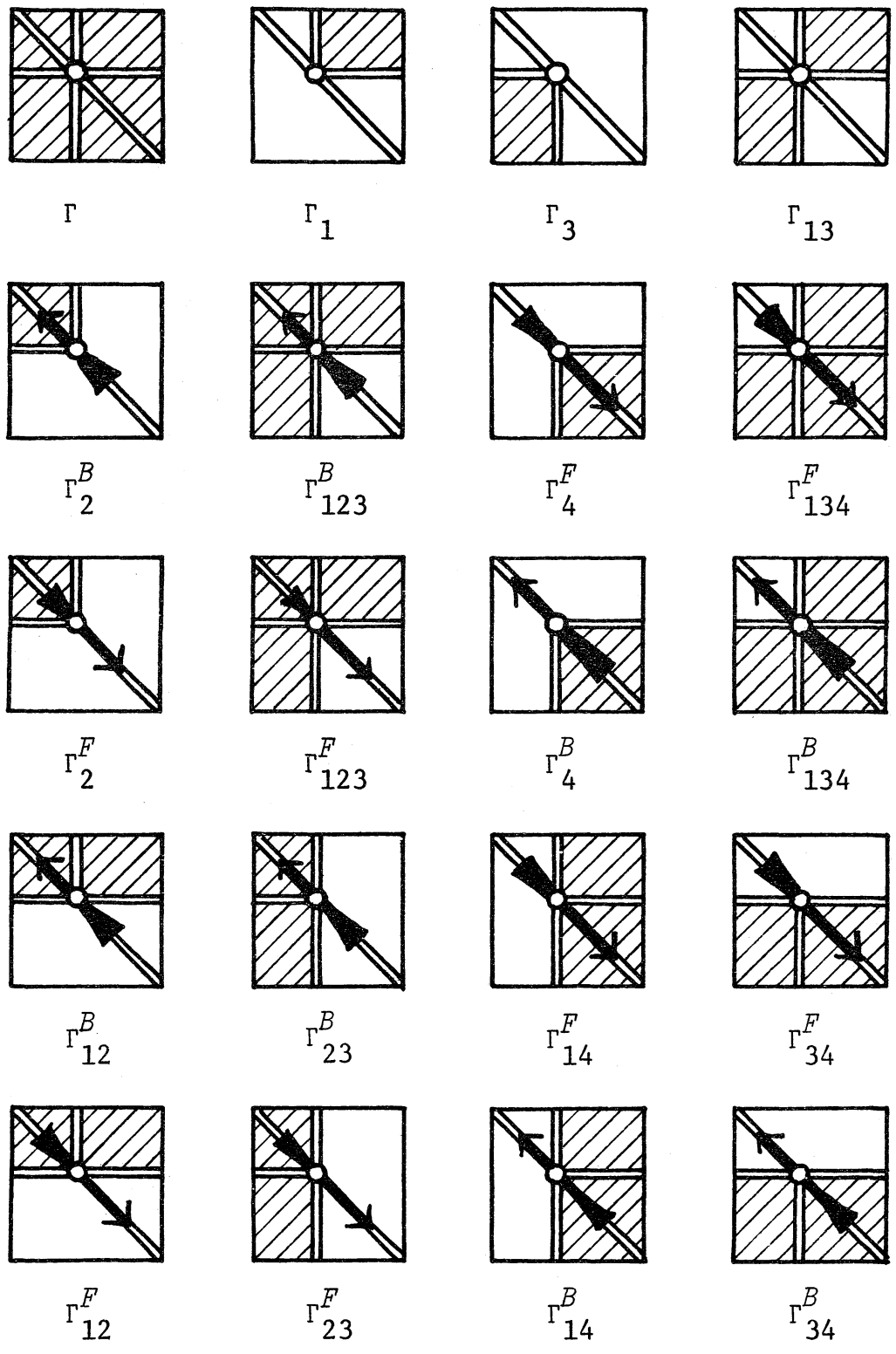
Γ
Γ_1
Γ_3
Γ_{13}
Γ^B_2
Γ^B_{123}
Γ^F_4
Γ^F_{134}
Γ^F_2
Γ^F_{123}
Γ^B_4
Γ^B_{134}
Γ^B_{12}
Γ^B_{23}
Γ^F_{14}
Γ^F_{34}
Γ^F_{12}
Γ^F_{23}
Γ^B_{14}
Γ^B_{34}

Fig. 1

and

$$\Gamma_2^B < \Gamma_2^F \quad \text{if the underlying network is complete.} \qquad (3.9)$$

(For further details, see [2].)

Some of these operators are characterized in terms of the corresponding path-sets as follows.

$\rho^{\varepsilon}(\Gamma_1)$ *consists of all the elementary paths of lengths* 0 *and* 1 *and of the paths* $(i_0, i_1, \ldots, i_{r-1}, i_r)$*'s such that* $i_0 < i_1 < \ldots < i_{r-1} < i_r$.

$\rho^{\varepsilon}(\Gamma_2^B)$ *consists of all the elementary paths of lengths* 0 *and* 1 *and of the paths* $(i_0, i_1, \ldots, i_{r-1}, i_r)$*'s such that* $i_t > i_0$ *and* $i_t > i_r$ *for* $t = 1, \ldots, r-1$.

$\rho^{\varepsilon}(\Gamma_3)$ *consists of all the elementary paths of lengths* 0 *and* 1 *and of the paths* $(i_0, i_1, \ldots, i_{r-1}, i_r)$*'s such that* $i_0 > i_1 > \ldots > i_{r-1} > i_r$.

$\rho^{\varepsilon}(\Gamma_4^F)$ *consists of all the elementary paths of lengths* 0 *and* 1 *and of the paths* $(i_0, i_1, \ldots, i_{r-1}, i_r)$*'s such that* $i_t < i_0$ *and* $i_t < i_r$ *for* $t = 1, \ldots, r-1$.

3.6. *Variants of Warshall-Floyd algorithm.* The path-set $\rho^{\varepsilon}(\langle {}^{k}_{N\times N} \rangle)$ consists of all the elementary paths of lengths 0 and 1 and of the paths of the form (i, k, j) $(i, j = 1, 2, \ldots, n)$.

The path-set $\rho^{\varepsilon}(\langle {}^{k_1}_{N\times N}\rangle \cdots\cdots \langle {}^{k_s}_{N\times N}\rangle)$ includes all the elementary paths $(i, i_1, \ldots, i_{r-1}, j)$'s $(r = 1, 2, \ldots, s+1;\ i_1, \ldots, i_{r-1} = k_1, \ldots, k_s;\ i, j = 1, 2, \ldots, n)$. Therefore,

$$\Gamma \triangleq \prod_{k\in N} \langle {}^{k}_{N\times N}\rangle \tag{3.10}$$

has its unique meaning as an operator valid for a complete network. This operator Γ represents the Warshall-Floyd algorithm [4] which has been referred to in §2.

From §3.6 we have various equivalent expressions for Γ such as follows.

$$\begin{aligned}
\Gamma &\equiv \Gamma^F_{123}\cdot\Gamma^F_4 \equiv \Gamma^B_{123}\cdot\Gamma^F_4 \equiv \Gamma^B_{134}\cdot\Gamma^B_2 \equiv \Gamma^F_{134}\cdot\Gamma^B_2 \\
&\equiv \Gamma^F_{12}\cdot\Gamma^F_{34} \equiv \Gamma^B_{12}\cdot\Gamma^F_{34} \equiv \Gamma^B_{14}\cdot\Gamma^B_{23} \equiv \Gamma^F_{14}\cdot\Gamma^B_{23} \\
&\equiv \Gamma^F_{23}\cdot\Gamma^F_{14} \equiv \Gamma^B_{23}\cdot\Gamma^F_{14} \equiv \Gamma^B_{34}\cdot\Gamma^B_{12} \equiv \Gamma^F_{34}\cdot\Gamma^B_{12} \\
&\equiv \Gamma^F_2\cdot\Gamma_{13}\cdot\Gamma^F_4 \equiv \Gamma^B_2\cdot\Gamma_{13}\cdot\Gamma^F_4 \equiv \Gamma^B_4\cdot\Gamma_{13}\cdot\Gamma^B_2 \equiv \Gamma^F_4\cdot\Gamma_{13}\cdot\Gamma^B_2 \\
&\equiv \Gamma_{13}\cdot\Gamma^B_2\cdot\Gamma^F_4 \equiv \Gamma_{13}\cdot\Gamma^F_4\cdot\Gamma^B_2 \\
&\equiv \Gamma_1\cdot\Gamma^B_2\cdot\Gamma_3\cdot\Gamma^F_4 \equiv \Gamma_3\cdot\Gamma^B_2\cdot\Gamma_1\cdot\Gamma^F_4 \\
&\equiv \Gamma_1\cdot\Gamma^F_4\cdot\Gamma_3\cdot\Gamma^B_2 \equiv \Gamma_3\cdot\Gamma^F_4\cdot\Gamma_1\cdot\Gamma^B_2 \ .
\end{aligned} \tag{3.11}$$

These expressions for Γ will be useful to devise less complex algorithms in the case where the underlying network is of special structure (cf. §4).

3.7. *Another proof of the validity of Katayama-Watanabe algorithm.* Table 1 shows that

$$\left\langle {}^{X}_{i\ j} \right\rangle \triangleq \prod_{k \in X} \left\langle {}^{k}_{i\ j} \right\rangle$$

has its unique meaning as an operator. It is sometimes convenient to adopt the following subsets of N as the X in the above operator $\left\langle {}^{X}_{i\ j} \right\rangle$.

$$\left.\begin{aligned} I(i, j) &= \{k \in N \mid i \leqq k \leqq j\} , \\ II(i, j) &= \{k \in N \mid i \leqq k, \ \ j \leqq k\} , \\ III(i, j) &= \{k \in N \mid j \leqq k \leqq i\} , \\ IV(i, j) &= \{k \in N \mid k \leqq i, \ \ k \leqq j\} . \end{aligned}\right\} \tag{3.12}$$

In the same way as we defined Γ-operators in §3.6, we define

$$\left.\begin{aligned} \gamma^{F}_{r} &\triangleq \gamma^{F(n)}_{r} \cdots\cdots \gamma^{F(1)}_{r}, \quad \gamma^{F(i)}_{r} \triangleq \left\langle {}^{R(i,n)}_{i\ \ n} \right\rangle \cdots\cdots \left\langle {}^{R(i,1)}_{i\ \ 1} \right\rangle , \\ \gamma^{B}_{r} &\triangleq \gamma^{B(1)}_{r} \cdots\cdots \gamma^{B(n)}_{r}, \quad \gamma^{B(i)}_{r} \triangleq \left\langle {}^{R(i,1)}_{i\ \ 1} \right\rangle \cdots\cdots \left\langle {}^{R(i,n)}_{i\ \ n} \right\rangle , \end{aligned}\right\} \tag{3.13}$$

where we set $r = 1, 2, 3$ or 4, according as $R = I, II, III$ or IV, respectively. We sometimes use symbols such as γ^{F}_{rs}, γ^{B}_{rs} or γ^{F}_{rst}, γ^{B}_{rst} to mean the product operators of $\left\langle {}^{R(i,j) \cup S(i,j)}_{i\qquad\ j} \right\rangle$'s or $\left\langle {}^{R(i,j) \cup S(i,j) \cup T(i,j)}_{i\qquad\ \ j} \right\rangle$'s, whose rigorous definition will be evident. For example, we may make use of

$$\gamma^{B}_{2}, \quad \gamma^{F}_{2}, \quad \gamma^{B}_{4}, \quad \gamma^{F}_{4},$$

$$\gamma^{B}_{1}, \quad \gamma^{F}_{1}, \quad \gamma^{B}_{3}, \quad \gamma^{F}_{3}, \quad \gamma^{B}_{13}, \quad \gamma^{F}_{13},$$

$$\gamma_2^B,\quad \gamma_2^F,\quad \gamma_{12}^B,\quad \gamma_{12}^F,\quad \gamma_{23}^B,\quad \gamma_{23}^F,\quad \gamma_{123}^B,\quad \gamma_{123}^F,$$

$$\gamma_4^B,\quad \gamma_4^F,\quad \gamma_{34}^B,\quad \gamma_{34}^F,\quad \gamma_{14}^B,\quad \gamma_{14}^F,\quad \gamma_{134}^B,\quad \gamma_{134}^F.$$

By an elementary computation we can derive various formulae connecting Γ-operators and γ-operators. For example, we have (see [2])

$$\left.\begin{aligned} &\gamma_1^B \equiv \gamma_1^F \equiv \Gamma_1, \qquad \gamma_{13}^B \equiv \gamma_{13}^F \equiv \Gamma_{13};\\ &\gamma_2^F \equiv \Gamma_2^F, \qquad\qquad \gamma_2^B \equiv \Gamma_2^B;\\ &\gamma_{12}^B \equiv \Gamma_2^B \cdot \Gamma_1, \end{aligned}\right\} \tag{3.14}$$

and

$$\gamma_{12}^F > \Gamma_1 \cdot \Gamma_2^F, \qquad \gamma_{12}^F \not< \Gamma_{12}^F, \qquad \gamma_{12}^B > \Gamma_{12}^B$$

if the underlying network is complete. (3.15)

The validity of Katayama-Watanabe algorithm can be proved on the basis of the preceding relations among operators as follows. Their algorithm is described in terms of γ-operators as $\gamma_{13}^F \cdot \gamma_2^B \cdot \gamma_4^F$, and, by virtue of (3.14), this is equivalent to $\Gamma_{13}^F \cdot \Gamma_2^B \cdot \Gamma_4^F$. However, we have already obtained in §3.6 this last expression as a variant of Warshall-Floyd algorithm Γ.

The validity of other methods mentioned in §2 may be proved in a similar way (see [2]), and the process of proof is far simpler than the direct proof given in the original papers.

4. *Optimal Algorithms for Specially Structured Networks*

4.1. *A general lower bound for the complexity of algorithms.*

In order to discuss the optimality of algorithms for specially structured networks, we establish the following preliminary theorems.

Theorem 7. *If a path* $p = (i, k_1, \ldots, k_{r-1}, j)$ $(r \geqq 2)$ *belongs to the path-set* $\rho^{\varepsilon}(\tau)$ *for an algorithm* τ, *then there exists at least one node* k_t $(1 \leqq t \leqq r-1)$ *such that* $\left\langle {}^{k_t}_{i\ j} \right\rangle$ *appears at least once in* τ.

Proof: Suppose none of $\left\langle {}^{k_1}_{i\ j} \right\rangle, \ldots, \left\langle {}^{k_{r-1}}_{i\ j} \right\rangle$ appear in τ. Then p would not belong to $\rho^{\varepsilon}(\tau)$, contrary to the assumption. ■

We say that two elementary paths $p = (i, k_1, \ldots, k_{r-1}, j)$ $(r \geqq 2)$ and $q = (i, h_1, \ldots, h_{s-1}, j)$ $(s \geqq 2)$ are *disjoint* if nodes $k_1, \ldots, k_{r-1}$ are not on q and $h_1, \ldots, h_{s-1}$ are not on p. For two distinct nodes i and j we denote by $u(i, j)$ the maximum number of disjoint elementary conductive paths of length greater than 1 from i to j. From Theorem 7 we have

Theorem 8. *Let* i *and* j *be arbitrary distinct nodes and* $p_1, p_2, \ldots, p_{u(i, j)}$ *be disjoint elementary conductive paths from* i *to* j *in a network* G. *If* τ *is a valid algorithm for* G, *then for each* p_v $[v = 1, 2, \ldots, u(i, j)]$ *there is at least*

one k_v *(distinct from* i *and* j*) on* p_v *such that* $\langle {}^{k_v}_{i\ j} \rangle$ *appears at least once in* τ.

and

Theorem 9. *Any valid algorithm for a network* G *contains at least* $\sum_{i \neq j} u(i, j)$ *occurrences of triple-operations, where the summation is taken over all the pairs of nodes.*

Remark 1. Note that, by virtue of a well-known theorem of the Menger type [12], the number $u(i, j)$ is equal to the minimum number of nodes whose removal separates nodes i and j from each other.

Remark 2. If G is complete, then $u(i, j) = n - 2$ for every pair (i, j), so that we have the lower bound $n(n-1)(n-2)$, in accordance with the arguments in the preceding section.

4.2. *Tree-structured networks.* For a partition of the set of nodes N of a network G into m disjoint subsets $N_1, N_2, \ldots, N_m$, we construct the graph $\tilde{G}$ which has $N_1, N_2, \ldots, N_m$ as the nodes and which has a branch connecting N_v and N_w $(v \neq w)$ if and only if there is a branch in G connecting a node of N_v and a node of N_w. If $\tilde{G}$ is a tree and if there is always a branch in G connecting nodes i and j when $i, j \in N_v$ or $i \in N_v$, $j \in N_w$ with N_v and N_w adjacent, we say that the network G is *tree-structured.* An

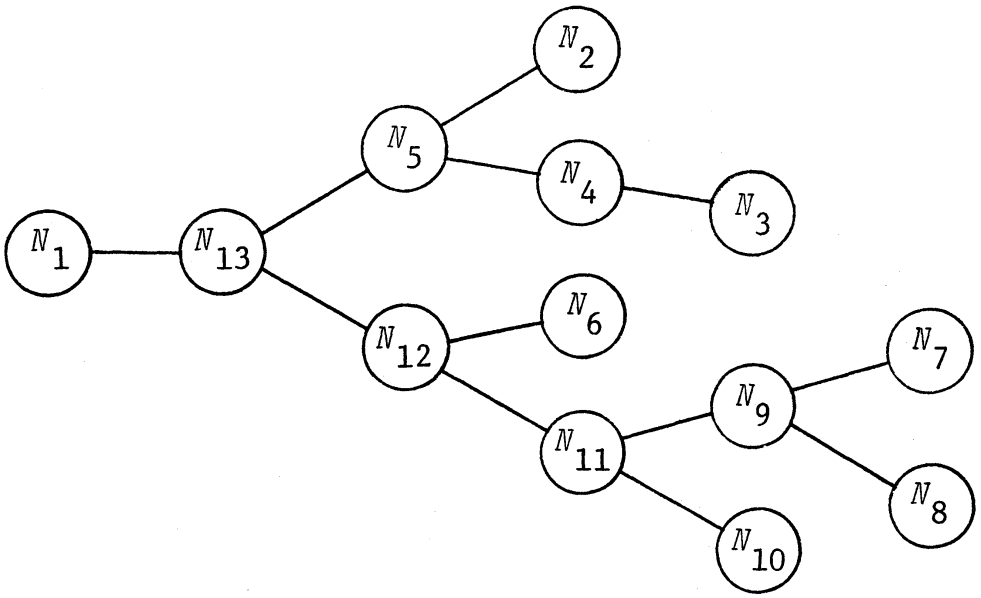

Fig. 2. $\tilde{G}_{tr}$ of a tree-structured network G_{tr}.

example of a tree-structured network is shown in Fig. 2.

Let G_{tr} be a tree-structured network. We say that two node sets N_v and N_w are *adjacent* in G_{tr} if nodes N_v and N_w in $\tilde{G}_{tr}$ are adjacent in $\tilde{G}_{tr}$. We say that a node set N_x is *intermediate* from N_v to N_w if every conductive path from a node in N_v to a node in N_w passes a node in N_x.

Theorem 8, when applied to tree-structured networks, gives

Theorem 10. *Let* τ *be a valid algorithm for a tree-structured network.*

(*a*) *If two distinct nodes* i *and* j *belong to the same node set* N_v, *then for every node* k *in* N_v *and all the node sets adjacent to* N_v *the triple-operation* $\langle {}_i{}^k{}_j \rangle$ *appears at least once in* τ.

(*b*) *If two node sets* N_v *and* N_w *are adjacent and* $i \in N_v$, $j \in N_w$, *then for every node* k *in* N_v *and* N_w *the triple-operation* $\langle {}^{k}_{i\ j} \rangle$ *appears at least once in* τ.

(*c*) *If two node sets* N_v *and* N_w *are not adjacent and* $i \in N_v$, $j \in N_w$, *then there exists an intermediate node set* N_x *such that for every node* k *in* N_x *the triple-operation* $\langle {}^{k}_{i\ j} \rangle$ *appears at least once in* τ.

Quite recently, D. R. Shier presented a method of solution for a tree-structured networks [13]. When we express this method in terms of triple-operations, we obtain an algorithm containing only those triple-operations which are mentioned in Theorem 10, each triple-operation occurring precisely once. Moreover, for nodes i and j ($i \in N_v$, $j \in N_w$, N_v and N_w not being adjacent) we can choose N_x in (*c*) of Theorem 10 such that $|N_x|$ is the minimum among all the N_x's intermediate from N_v to N_w (see §4.3). Thus, when appropriately programmed, his algorithm is optimal.

4.3. *Cascade networks.* As a special case of tree-structured networks, a network such as in Fig. 3 may be considered. However,

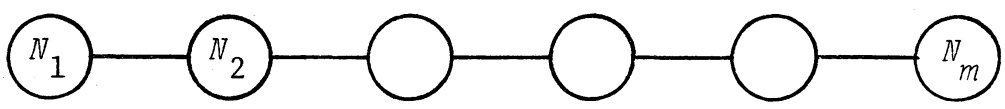

Fig. 3

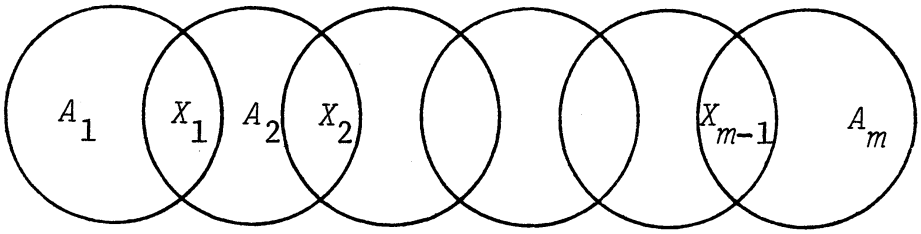

Fig. 4. A cascade network.

in the following we shall consider "cascade networks" such as illustrated in Fig. 4 by slightly generalizing it.

Let the set of nodes N be decomposed into disjoint subsets $A_1, X_1, A_2, X_2, \ldots, X_{m-1}, A_m$. For the sake of simplicity, we use notations such as $\overline{A}_p = X_{p-1} \cup A_p \cup X_p$, $A_p^- = X_{p-1} \cup A_p$ and $A_p^+ = A_p \cup X_p$ $(p = 1, 2, \ldots, m;\ X_0 = X_m = \emptyset)$. A *cascade network* is a network such that two nodes are connected by a branch if and only if they belong to one and the same $\overline{A}_p$. Distance matrices on a cascade network G_{cs} take the form in Fig. 5.

Let us apply theorems 8 and 9 to a cascade network G_{cs}. For nodes i and j in the same set $\overline{A}_p$ we have $u(i, j) = |\overline{A}_p| - 2$.

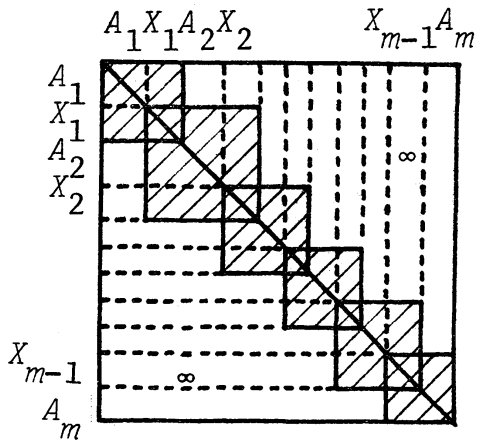

Fig. 5. A distance matrix corresponding to a cascade network.

For p and q such that $1 \leqq p < q \leqq m$ we define $\hat{y}(p, q)$ by $|X_{\hat{y}(p,q)}| = \min(|X_p|, \ldots, |X_{q-1}|)$. Then, for nodes $i \in A_p^-$ and $j \in A_q^+$ we have $u(i, j) = u(j, i) = \hat{y}(p, q)$. Therefore, the complexity of any valid algorithm for a cascade network G_{st} is at least

$$\sum_{p=1}^{m} \bar{a}_p(\bar{a}_p-1)(\bar{a}_p-2) - \sum_{p=1}^{m-1} x_p(x_p-1)(x_p-2) + 2 \sum_{p<q} a_p^- a_q^+ x_{\hat{y}(p,q)} , \qquad (4.1)$$

where $a_p = |A_p|$, $x_p = |X_p|$, $\bar{a}_p = |\bar{A}_p|$, $a_p^- = |A_p^-|$ and $a_p^+ = |A_p^+|$.

The following algorithm, proposed initially by us in [2], consists of only the above mentioned triple-operations, each occurring precisely once and, therefore, is optimal. This algorithm is a modified form of $\Gamma_{13} \cdot \Gamma_2^B \cdot \Gamma_4^F$ in §3.6.

Let us denote the intersection of diagonal blocks with $R(k)$ by $R(k; D)$, where $R = I$, II, III or IV (see Fig. 6). We define $\Gamma_{2(\text{in})}^B$, $\Gamma_{4(\text{in})}^F$ and $\Gamma_{13(\text{in})}$ as operators obtained by replacing $\left\langle {}^{k}_{R(k)} \right\rangle$'s in Γ_2^B, Γ_4^F and Γ_{13}, respectively, with the corresponding $\left\langle {}^{k}_{R(k;D)} \right\rangle$'s. Sor far as G_{cs} is concerned, it is readily seen that

$$\left\langle {}^{k}_{II(k)} \right\rangle \equiv \left\langle {}^{k}_{II(k;D)} \right\rangle , \qquad \left\langle {}^{k}_{IV(k)} \right\rangle \equiv \left\langle {}^{k}_{IV(k;D)} \right\rangle ,$$

because the values outside the diagonal blocks remains unchanged (i.e. remains to be ∞) under these operators. Therefore, the

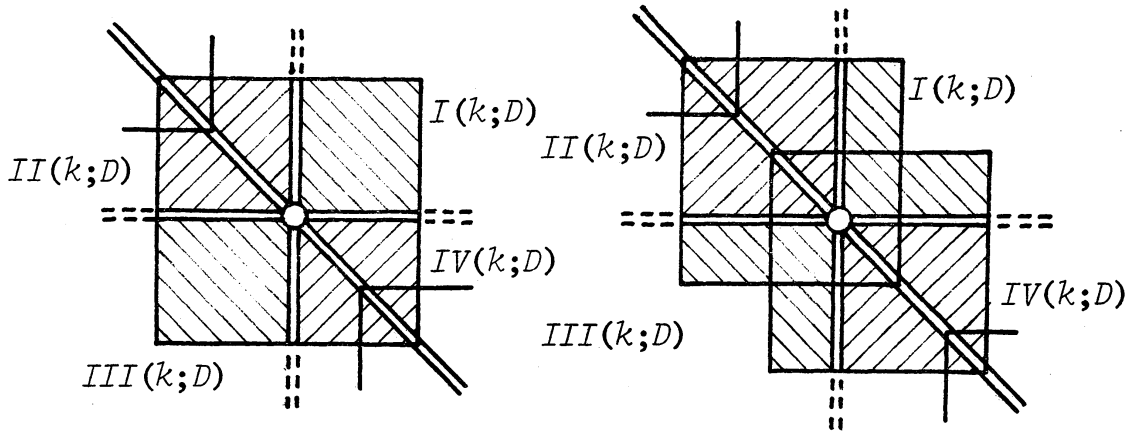

Fig. 6. Illustration of $R(k; D)$'s ($R = I$, II, III or IV).

effect of Γ_2^B and Γ_4^F in $\Gamma_{13}\cdot\Gamma_2^B\cdot\Gamma_4^F$ is equal to that of $\Gamma_{2(\mathrm{in})}^B$ and $\Gamma_{4(\mathrm{in})}^F$, respectively. Note, also, that the values outside the diagonal blocks have no influence upon the diagonal blocks when we apply the algorithm Γ_{13}. Thus, the algorithm $\Gamma_{13(\mathrm{in})}\cdot\Gamma_{2(\mathrm{in})}^B\cdot\Gamma_{4(\mathrm{in})}^F$ computes the correct value of d_{ij}^{∞}'s in the diagonal blocks.

Next, let $i \in A_p^-$, $j \in A_q^+$ and $p < q$. Then for any $y(p, q)$ such that $p \leqq y(p, q) < q$, we have

$$d_{ij}^{\infty} = \min_{k \in X_{y(p,q)}} (d_{ik}^{\infty} + d_{kj}^{\infty}) , \tag{4.2}$$

because every elementary conductive paths from i to j passes $X_{y(p,q)}$. This fact implies that the values d_{ij}^{∞}'s outside the diagonal blocks may be computed in an increasing order of $q - p$ ($i \in A_p^-$, $j \in A_q^+$ or $j \in A_p^-$, $i \in A_q^+$; $p < q$). Thus, we have the following algorithm, where we set $y(p, q) = \hat{y}(p, q)$ in

order to make the algorithm optimal.

$$\left.\begin{aligned} \Gamma_{13(\mathrm{out})} &\triangleq \Gamma_{13}^{m-1}\cdot\Gamma_{13}^{m-2}\cdots\cdots\Gamma_{13}^{2}\cdot\Gamma_{13}^{1}\ , \\ \Gamma_{13}^{r} &\triangleq \prod_{q-p=r}\Gamma_{13}^{p,q} \qquad (r = 1, 2, \ldots, m-2, m-1), \\ \Gamma_{13}^{p,q} &\triangleq \prod_{k\in X_{\hat{y}}(p,q)} \left\langle A_p^- x A_q^+ \bigcup^{k} A_q^+ x A_p^- \right\rangle \qquad (p < q)\ . \end{aligned}\right\} \tag{4.3}$$

As we have already mentioned, this algorithm $\Gamma_{13(\mathrm{out})}\cdot\Gamma_{13(\mathrm{in})}\cdot\Gamma_{2(\mathrm{in})}^{B}\cdot\Gamma_{4(\mathrm{in})}^{F}$ contains triple-operations of the number equal to (4.1) so that it is optimal.

If $a_1 = a_2 = \ldots = a_m = a$ and $x_1 = x_2 = \ldots = x_{m-1} = x$, then the leading term (i.e. the terms of the order highest in a and x) of (4.1) is equal to

$$ma^3 + (m^2+5m-6)a^2x + (2m^2+6m-14)ax^2 + (m^2+2m-7)x^3\ . \tag{4.4}$$

The leading term of the complexity of the algorithm in [14] or that in [15] for cascade networks is equal, respectively, to

$$(2m-1)a^3 + (m^2+11m-15)a^2x + (2m^2+18m-35)ax^2 + (m^2+11m-23)x^3\ , \tag{4.5}$$

and

$$ma^3 + (m^2+6m-7)a^2x + (2m^2+10m-20)ax^2 + (m^2+6m-14)x^3\ . \tag{4.6}$$

4.4. *Star networks.* A star network is another special case of tree-structured networks.

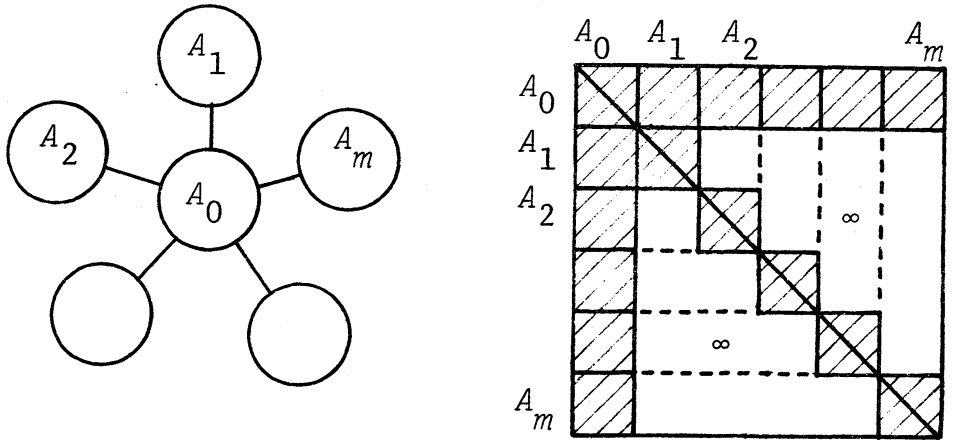

Fig. 7. A star network and its distance matrix.

If the set of nodes N of a network G_{st} is decomposed into disjoint subsets $A_0, A_1, A_2, \ldots, A_m$, and if there is no branch connecting nodes i and j not belonging to the same subset $\overline{A}_p = A_0 \cup A_p$, we call the network a *star network*. An example of G_{st} with its distance matrix is shown in Fig. 7.

Theorem 10 shows that a triple-operation $\langle {}^{k}_{i\ j} \rangle$ with pivot $k \in A_p$ must appear at least once for every node-pair (i, j) in $\overline{A}_p \times \overline{A}_p$ $(p = 1, 2, \ldots, m)$, and a triple-operation $\langle {}^{k}_{i\ j} \rangle$ with pivot $k \in A_0$ must appear at least once for every node-pair (i, j) in $N \times N$. Thus the complexity of any valid algorithm for G_{st} is bounded from below by

$$a_0(n-1)(n-2) + \sum_{p=1}^{m} a_p(\overline{a}_p-1)(\overline{a}_p-2) , \qquad (4.7)$$

where $a_p = |A_p|$ and $\overline{a}_p = |\overline{A}_p|$ $(p = 0, 1, 2, \ldots, m)$.

Let us apply the Warshall-Floyd algorithm for G_{st}, omitting all those operations ineffective by virture of the structure of

G_{st}. Since only the submatrix $\overline{A}_p \times \overline{A}_p$ is transformed by $\langle {}^{k}_{i\ j} \rangle$ when $k \in A_p$ $(p = 1, 2, \ldots, m)$, we have

$$\left\langle {}^{k}_{N\times N} \right\rangle \equiv \left\langle {}^{k}_{\overline{A}_p \times \overline{A}_p} \right\rangle \quad \text{if } k \in A_p \ (p = 1, 2, \ldots, m).$$

Therefore, the algorithm

$$\Gamma_{st} \triangleq \Big(\prod_{k\in A_p} \left\langle {}^{k}_{N\times N} \right\rangle \Big) \cdot \prod_{p=1}^{m} \Big(\prod_{k\in A_p} \left\langle {}^{k}_{\overline{A}_p \times \overline{A}_p} \right\rangle \Big)$$

is valid for G_{st}. Moreover, the complexity of Γ_{st} is equal to (4.7). Thus Γ_{st} is optimal. If $a_1 = a_2 = \ldots = a_m = a$, then the leading term of (4.7) is equal to

$$a_0 n^2 + ma(a_0 + a)^2 . \tag{4.8}$$

References

[1] M. Iri, *Shortest-Path Algorithms on a Network: A Survey* (in Japanese, 回路上で最短距離を求める算法総合報告), Mimeographed Note, June 1968.

[2] M. Iri and M. Nakamori, "Path-Sets, Operator Semigroups and Shortest-Path Algorithms on a Network," *RAAG Research Notes*, 3*rd Series*, No. 185 (October 1972).

[3] M. Nakamori, "A Note on the Optimality of Some All-Shortest-Path Algorithms," *J. Operations Res. Soc. Japan*

15 (1972), 201-204.

[4] R. W. Floyd, "Algorithm 97: Shortest Path," *Comm. ACM* 5 (1962), 345.

[5] G. B. Dantzig, "All-Shortest-Routes in a Graph," *Théorie des Graphes*, pp. 91-92, Dunod, Paris, 1968.

[6] H. Katayama and H. Watanabe, "Shortest-Path Problems in a Communication Network" (in Japanese, 通信網における最短路問題), 1968 *Joint Convension Record of Four Institutes of Electrical Engineers of Japan* (昭和43年度電気四学会連合大会講演論文集), Vol. 1 (Basic Theory), No. 47, pp. 56-57, March 1968.

[7] B. A. Farbey, A. H. Land and J. D. Murchland, "The Cascade Algorithms for Finding All Shortest Distances in a Directed Graph," *Management Science* 14 (1967), 19-28.

[8] T. C. Hu, "Revised Matrix Algorithms for Shortest Paths," *SIAM J. App. Math.* 15 (1967), 207-218.

[9] M. Pollack and W. Wiebenson, "Solution of the Shortest-Route Problem: A Review," *Operations Research* 8 (1960), 224-230.

[10] A. J. Hoffman and S. Winograd, "Finding All Shortest Distances in a Directed Network," *IBM. J. Res. Develop.* 16 (1972), 412-414.

[11] T. C. Hu, "A Decomposition Algorithm for Shortest-Paths in a Network," *Operations Research* 16 (1968), 91-102.

[12] M. Iri, *Network Flow, Transportation and Scheduling: Theory and Algorithms*, Academic Press, New York, 1969.

[13] D. R. Shier, "A Decomposition Algorithms for Optimality Problems in Tree-Structured Networks," *Discrete Mathematics* 6 (1973), 175-189.

[14] T. C. Hu and W. T. Torres, "Shortcut in the Decomposition Algorithm for Shortest Paths in a Network," *IBM J. Res. Develop.* 13 (1969), 387-390.

[15] J. Y. Yen, "On Hu's Decomposition Algorithm for Shortest-Paths in a Network," *Operations Research* 19 (1971), 983-985.